Integrated Demonstration of Instrument Placement, Robust Execution and Contingent Planning

L. Pedersen[1], M. Bualat[2], D. Lees[1], D.E. Smith[2], R. Washington[3]
NASA Ames Research Center, Moffett Field, CA 94035-1000
[1]QSS Group, Inc at NASA ARC, [2]NASA ARC, [3]RIACS at NASA ARC
{pedersen, lees, de2smith, richw}@email.arc.nasa.gov, Maria.G.Bualat@nasa.gov

Keywords rovers, instrument deployment, rover operations, K9, contingent planning, robust execution, Viz, data visualization

Abstract

This paper describes an integrated demonstration of ground-based contingent planning, robust execution and autonomous instrument placement for the efficient exploration of a site by a prototype Mars rover.

1.1. Introduction

Approaching science targets, such as rocks, and placing instruments against them to take measurements is the raison d'être of a planetary surface exploration rover, such as the planned 2009 Mars Science Laboratory (MSL) rover. This is necessary to acquire samples, determine mineralogy, obtain microscopic images and perform other operations needed to understand the planet's geology and search for evidence of past or present life. Significant science cannot be done with remote measurements only.

Figure 1 K9 Rover autonomously places a microscopic camera to examine a rock target in the NASA Ames Marscape.

In order to accomplish the task of instrument placement within a single cycle with the robustness required for a mission, the on-board software must be able to handle failures and uncertainties encountered during the component tasks. A task may fail, requiring recovery or retrying. Tasks may exhibit a high degree of variability in their resource usage, using more (or less) time and energy than expected. Finally, the state of the world and the rover itself may be predictable only to a limited extent. Exploring multiple rock targets further exacerbates this situation. These factors require that the rover's software have the ability to reason about a wide range of possible situations and behaviors. A simple script is insufficient; instead, the rover can use either on-board task planning or off-board planning in conjunction with robust on-board execution.

In November and December of 2002, researchers at NASA Ames Research Center (ARC) successfully demonstrated an end-to-end single cycle instrument deployment scenario from ground operations and planning to execution and science data capture. This demonstration occurred in the newly constructed Marscape rover test site.

The demonstration mission scenario begins with the NASA Ames K9 rover at the site to be explored (a simulated lakebed in the Marscape). K9 acquires a panoramic set of stereo images of the site. These are downloaded and processed off-board to create a virtual terrain model of the environment, rendered in the Ames Viz 3D virtual reality interface. The mission controllers explore the virtual world and choose rock targets of interest. For each rock they decide where the rover needs to be to examine it, what measurements are desired at that location, and how much these measurements are worth. In addition, they specify the allowed paths between the various locations, including the start position and any additional decision points. This information is automatically saved to a file for use by the mission planner.

Our goal is for the rover to obtain, in a single command cycle, the set of measurements that maximize the expected utility subject to constraints on time, power consumption and where it can go. We accomplish this using an off-board mixed initiative contingent planner along with robust on-board execution. This is consistent with current mission practice, which requires intensive sequence verification before uplink. In addition, the perceived additional risk of an on-board planner could delay acceptance by mission managers.

The planner generates a sequence, with contingencies, that is uploaded to the rover, where it is executed by the conditional executive. The rover navigates, via decision points, to rock targets where it stops and autonomously places an arm mounted microscopic camera against the target and acquires a measurement.

The remainder of this paper describes the various components of this demonstration: the K9 rover, the Marscape test facility, the science interface (Viz), the contingent planner, the conditional executive, and the instrument placement system.

1.2. Robotic Testbed and Outdoor Test Facility

1.3. K9 Rover

The K9 Rover is a 6-wheel steer, 6-wheel drive rocker-bogey chassis outfitted with electronics and instruments appropriate for supporting research relevant to remote science exploration. The main CPU is a 750 MHz PC104+ Pentium III running the Linux operating system. An auxiliary microprocessor communicates with the main CPU over a serial port and controls power switching and other I/O processing. The motion/navigation system consists of motor controllers for the wheels and pan/tilt unit, a compass, and an inertial measurement unit.

The K9 rover software architecture uses the Coupled Layered Architecture for Robotic Autonomy (CLARAty)[9] developed at JPL, in collaboration with ARC and Carnegie Mellon University. By developing our instrument placement technology under the CLARAty architecture, we can easily port the system to other CLARAty robots.

1.3.1. K9 Cameras

Figure 2 K9 camera systems. Mast-mounted navigational and science cameras (left) and front hazard avoidance cameras (right).

K9 is equipped with three camera pairs: a front-mounted forward looking pair of b/w stereo hazard cameras and mast mounted stereo pairs of high resolution color science cameras and wide field of view b/w navigation cameras (Figure 2). The navigation and science stereo pairs are mounted on a common pan-tilt unit such that they can acquire image panoramas from around the rover.

The hazard cameras overlook the arm workspace. Being fixed, and close to the target area, they are the easiest to calibrate with respect to the arm, and are used to build 3D models of the workspace prior to placing an instrument.

1.3.2. Instrument Arm

Figure 3 K9 5 DOF arm deployed.

K9's instrument arm (Figure 3) is a 5-DOF robotic manipulator based on a 4 DOF FIDO MicroArm IIA design from JPL[11]. It is approximately 5.0 kg with a total extended length of 0.79 meters. The waist yaw, shoulder pitch, elbow pitch, forearm twist (designed at Ames), and wrist pitch joints of the arm allow arbitrary x-y-z instrument placement as well as pitch and yaw control within the arm workspace. These rotational aluminum joints are connected by graphite epoxy tube links. The links are configured in a side-by-side orientation, with the two links running directly next to each other.

The payload mass for K9's arm is estimated to be about 1.5 kg (3.3 lbs) with a strong-arm lifting capacity of about 2.5 kg (5.5 lbs) when fully extended in the horizontal position. Each joint in the arm has an embedded MicroMo 1319 series motor with an integrated planetary gear head and magnetic encoder. (Additional harmonic drive gearing was needed past the actuator to meet the significant torque requirements.) The no-load output speed varies from joint to joint, but averages about 0.1 radians per second. External to each joint is a multi-turn potentiometer that is coupled to the rotor and is used for initial arm calibration. The calibration procedure and magnetic encoders result in a positional accuracy of +/- 2 mm.

1.3.3. CHAMP Microscopic Camera

Affixed at the end of K9's arm is the CHAMP (Camera Hand-lens MicroscoPe) microscopic camera [8] (Figure 4). It has a movable CCD image plane, allowing it to obtain focused images over a wide depth of field, from a few millimeters up to several meters.

Because rotation about CHAMP's long axis does not need to be controlled, placing CHAMP flat against a rock requires control of five degrees of freedom. K9's arm has a full 5 degrees of freedom, removing the need to coordinate simultaneous arm

and rover base motion. The rover's base only needs to move to within arm's reach of the rock, and can remain stationary during arm movement.

CHAMP has three spring-loaded mechanical distance sensors around its face that report contact with the rock. CHAMP can acquire a Z-stack of images from a target, each focused at a slightly different depth. These can be combined into a composite focused image (Figure 14).

Figure 4 CHAMP camera deployed onto rock.

1.4. Marscape

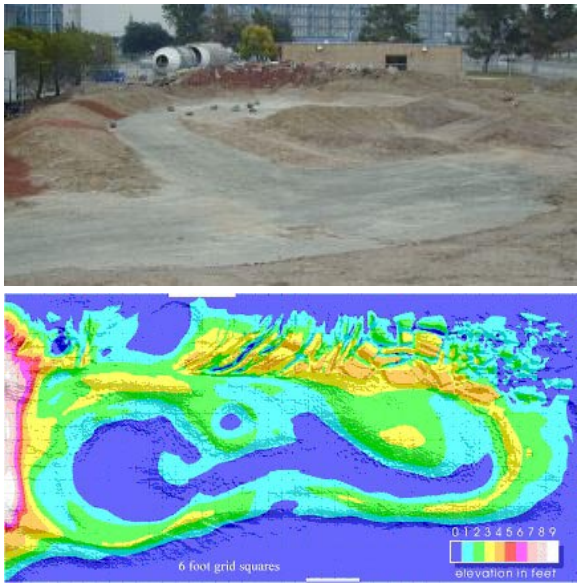

Figure 5 Marscape view toward dry lakebed and impact crater (top); Marscape elevation map showing streambed, delta, lakebed, volcano and chaotic terrain (bottom).

The integrated demonstration took place in the Marscape (Figure 5), a 3/4-acre rover test site built at Ames. Marscape is unique in that it features a richly varied topography with many occluded areas and consistent with those aspects of the Martian environment and geology of greatest scientific interest. Marscape's design includes a dry streambed with exposed sedimentary layers. The streambed drains into a dry lakebed with evaporite deposits. Overlapping the lakebed is an old meteorite impact crater, partially broken in a manner consistent with past water erosion. A volcanic zone (left in elevation map view) intruding onto the lakebed gives rise to an area of past hydrothermal activity, including hot springs, known to be excellent sites for the preservation of evidence for life. In the top part of the image we see a basalt magmatic cap has intruded over the sedimentary environment, giving rise to a chaotic terrain.

A trailer provides power, wireless network coverage, and shelter for engineers monitoring operations.

1.5. Ground Operations

1.6. Scene Visualization and Activity Specification

Stereo imagery from K9 is downloaded to the Ground Operations station and processed by the "Ames stereo-pipeline" to generate accurate high-resolution 3D terrain models of the remote site using binocular disparity information. The terrain model is viewed and manipulated in a virtual reality system called "Viz" [10]. Viz is an interactive simulation environment equipped with a number of science analysis and operations tools. These include lighting, pose and viewpoint simulation (Figure 6). A suite of measurement tools is also provided that allows intuitive interrogation of the remote site, and Viz is integrated with a kinematic simulation engine, called "VirtualRobot" (Figure 6), that provides rover pose and viewpoint simulation capabilities for operations planning.

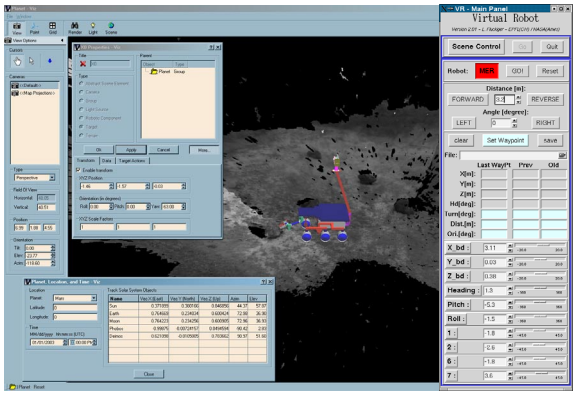

Figure 6 The Viz 3-D interface (left) and Virtual Robot control panel (right).

Using these tools, scientists specify a series of science targets and the paths and waypoints that connect them (Figure 7). For each target, the objectives are specified, along with the utility of each objective. This information is used to create an "operations file" which is post-processed to create an input file for the contingent planner (Section 3.2). The planner takes the list of desired science targets and utilities and creates a plan that maximizes science return subject to resource and safety constraints. We expect to do more work in the future

to "close the loop" between the planner and Viz (e.g. to visualize the results from the planner directly in the 3-D scene) so that scientists can get feedback about which targets will be visited and which will not and can adjust their plans accordingly.

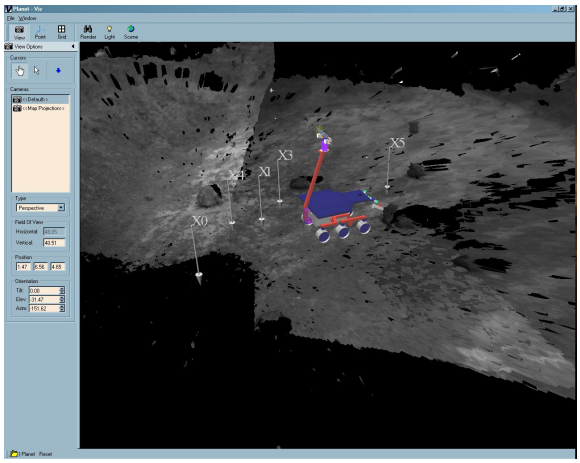

Figure 7 A set of science targets and waypoints.

1.7. Contingent Planning

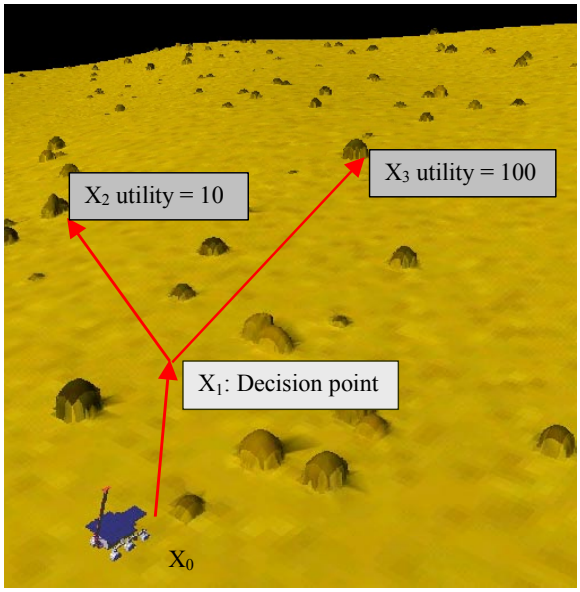

Figure 8 Waypoint and utility planning for instrument placement.

Once a set of science objectives has been chosen using Viz, these objectives, and their values are passed to a contingent planning system. This planning system determines which of the objectives to pursue along with the detailed commands necessary to achieve those objectives. In addition, it also inserts "contingency branches" into the plan to cover situations where the plan might possibly fail. In the example shown in Figure 8, suppose the planner initially constructs a plan to go to waypoint X_1, and then location X_3. It could then add a contingency branch to go to X_2 instead, if, upon arrival at X_1, there is not enough power or time available to continue to X_3.

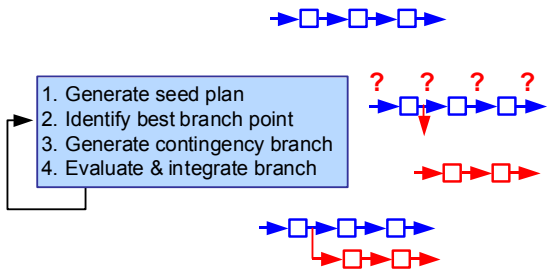

Figure 9 The Just-In-Case planning approach.

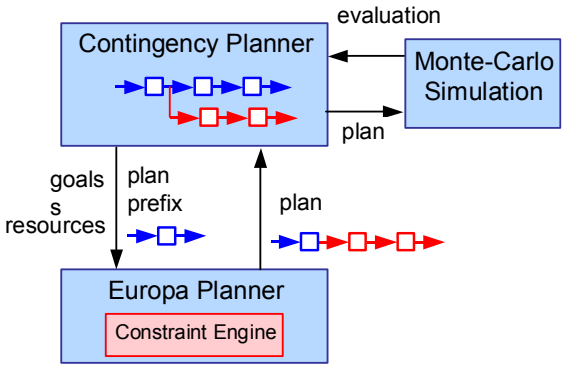

Figure 10 Architecture of the contingency planner.

This contingency planning is done using an incremental Just-In-Case approach [5], as illustrated in Figure 9. First a "seed" plan is generated having maximum expected utility. That is, the plan achieves the best objectives possible given the expected resources available (time and energy), and expected consumption of those resources by the actions involved. This plan is then evaluated to determine where it might fail, given uncertainty in resource consumption by the various actions involved. A branch point is then chosen, either by a user, or by using simple heuristics. An alternative, or *contingency*, plan is then constructed for this branch, and incorporated into the primary plan. The resulting conditional plan is again evaluated, and additional branches can be added as needed.

The architecture of the contingency planner is shown in Figure 10. The contingency planner makes use of the Europa planning engine [7][6] to generate seed plans, and to generate the plans for the contingency branches. For constructing a seed plan, the contingency planner gives Europa the goals, expected resource availability, and expected resource consumption of actions. When the plan comes back, the contingency planner evaluates it using Monte Carlo simulation to determine the impact of uncertainty in resource usage. The plan is then displayed in a JAVA GUI (Figure 11) for the user to examine. The user can select places where the planner should try to build a contingency branch. To build the branch, the planner passes appropriate goals, the prefix of the plan (prior to the branch point), and resource availability to Europa. In this

case, the expected resource availability is defined by the branch condition, because that bounds the resources that will be available if the branch is taken.

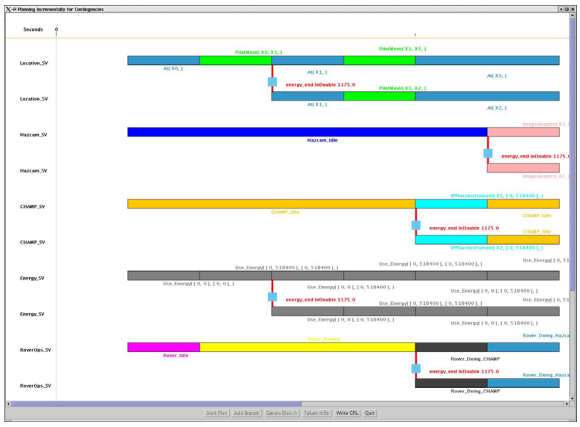

Figure 11 Planner GUI, displaying a set of branching timelines for different attributes like rover location and arm state.

The problem of automatically choosing good branch points and good branch conditions is quite hard in general (see [1], [4] for details). Intuitively, it might seem that a good place to put a contingency branch is at the place where the plan is most likely to fail. Unfortunately, this is often near the end of the plan, when resources (time and energy) are nearly exhausted. With few resources remaining, there may not be any useful alternative plans.

Instead, one would like to anticipate failures earlier in the plan, when useful alternatives remain. In other words, the planner is looking for the point(s) in the plan where a contingent branch could be added that would maximally increase the overall utility of the plan. In general, this quantity is very difficult to compute. In [4] we outline a heuristic technique for estimating both the expected gain of a given branch point, and the condition for that branch. This information can be used to do automatic branch point and condition selection. We have not yet completed the implementation and testing of this technique, but expect to incorporate it into the contingency planner in the near future.

1.8. Rover Operations

Once generated, the sequence with contingent branches is uplinked to K9.

1.9. Robust Execution

The CRL Executive is responsible for interpretation of the contingent plan coming from the ground and generated by the contingent planner. The CRL Executive is designed to be more capable than traditional sequence execution engines; it can handle the expressive plans generated by the contingent planner and can perform limited plan adaptation itself.

The planner translates its plan into the Contingent Rover Language (CRL) for uplink, and the CRL Executive interprets the CRL-encoded plan directly. CRL is a flexible, conditional sequence language that allows for execution uncertainty [2]. CRL expresses temporal constraints and state constraints on the plan, but allows flexibility in the precise time that actions must be executed. Constraints include conditions that must hold before, during, and after actions are executed. A recent addition to CRL is the ability to specify concurrent threads of activity.

A primary feature of CRL is its support for contingent branches to handle potential problem points or opportunities in execution. The contingent branches and the flexible plan conditions allow a single plan to encode a large family of possible behaviors, thus providing responses to a wide range of situations.

The structure of the CRL plan language and its interpretation are completely domain-independent. Domain-dependent information is added by specifying a command dictionary, with command names and argument types, and a command interface, which passes commands to the rover and return values and state information from the rover.

The CRL Executive is responsible for interpreting the CRL command plan coming from ground control, checking run-time resource requirements and availability, monitoring plan execution, and potentially selecting alternative plan branches if the situation changes. At each branch point in the plan, there may be multiple eligible options; the option with the highest expected utility is chosen. For this demonstration, the contingent planner generated mutually exclusive branches.

A novel feature of the CRL Executive is its support for “floating contingencies,” which are plan fragments that may be inserted at any point in execution [3]. For example, a plan to perform opportunistic science during a traverse is naturally expressed as a floating contingency, since the presence and position of an interesting science target is unknown before the traverse. Likewise, a plan to stop and recharge the battery is another example of a floating contingency. In general, floating contingencies would be impractical for the planner to consider because of the large number of possible branch points that they would add to a plan.

The CRL Executive is implemented as a multi-threaded, event-based system (see Figure 12). Around a central Executive event-processing loop are threads to handle timing, event monitoring, action execution monitoring, and telemetry gathering. The central event processor sends requests to the other threads (for example, "wake up at time 20" or "notify when battery state of charge is below 4Ah") and receives events relevant to those requests. This architecture allows the CRL Executive to support concurrent activities and flexible action conditions expressible within the CRL language.

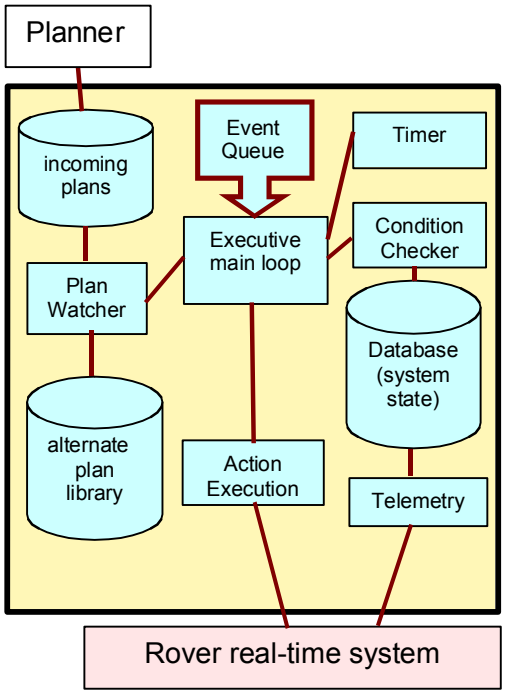

Figure 12 CRL Executive structure. The main event loop communicates with other threads for services such as timing, action monitoring, and event monitoring. External connections are to a planner, which supplies new plans to execute, and a rover real-time system, which executes actions and supplies telemetry data.

1.10. Target Approach and Instrument Placement

The uploaded CRL sequence commands K9 to move to rock targets, via waypoints. K9 stops at a target and assesses the rock scene in front of it to autonomously place CHAMP and acquire microscopic images. Figure 13 shows this sequence of activities in greater detail.

Currently, the rover uses deduced reckoning to maneuver to a location in front of the target. We do not yet use visual tracking, or other means, to maintain a fix on the rock target, or even a location on the rock, as the rover moves. Given our combined error budget of 30cm and navigation error of 5%, we can only work with targets within 3m distance of the rover.

Once the rover has moved up to the target, it must determine where to place the instrument, what pose is needed, and check that the target surface will permit the instrument to be placed there.

If Mission Control specified a particular final pose for the instrument, relative to a target that has been accurately tracked, then this task is unnecessary. Scientists at Mission Control might wish to specify an entire rock as a target, not just a given point. Not only is such over-specification unnecessary; it may over-constrain the problem, and might not even be feasible prior to the rover approaching close enough to the rock to see it in sufficient detail. Or it might simply not be possible to track a single point with enough precision. In these cases, scientists are compelled to request a measurement anywhere on a rock (or large area on it).

The first step in determining where to place an instrument on a rock target (or other large area) is to obtain a 3D scan of the work area. This can be done with the stereo hazard cameras, which have the best view of the work area and are the easiest to calibrate with respect to the arm.

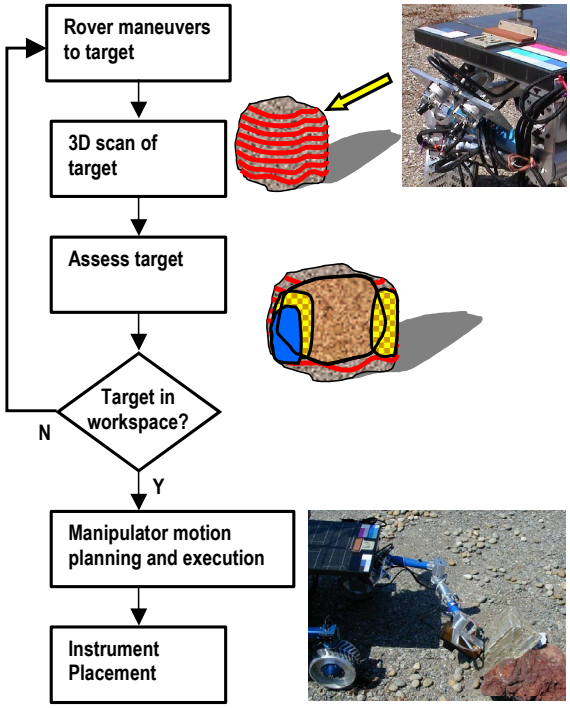

Figure 13 Instrument placement execution flow.

The rock (or target area) in the 3D model of the work area is segmented from the background using an iterative 3D clustering algorithm[11]. This algorithm is very robust to noise, requiring only that the ground be relatively flat (but at an arbitrary orientation) and that the work area have at most one rock significantly larger than any clutter in the scene.

All points on the target rock are checked for consistency with the rover instrument (CHAMP) to ensure that it does not get damaged during placement. The simplest check for each point is to find all points within a given radius, compute the best-fit plane, and check that maximum deviations do not exceed some preset tolerance.

Finally, the instrument can be placed. First, via a series of pre-planned waypoints the arm is un-stowed and put in a holding position. Next it goes to a pose near the highest priority target pose in the workspace, holding back a safe distance along the target surface normal. To compensate for possible small errors in surface location, the instrument's final approach is along the measured normal to the target

rock face, moving slowly forward until contact is confirmed by mechanical sensors.

Once the arm places the camera, we obtain microscopic images of the rock surface (Figure 14).

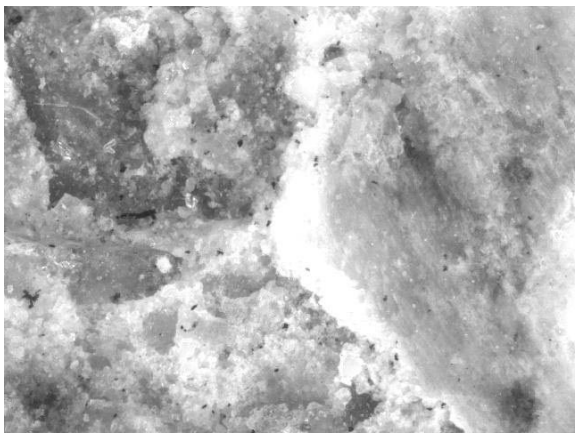

Figure 14 Composite microscopic image of target rock.

1.11. Summary

It has been speculated that the use of nuclear power to extend the 2009 Mars rover mission to 1000 days decreases the need for autonomy, as there would be sufficient time to accomplish measurements in the traditional, time consuming way, without having to risk autonomy. This is fallacious for several reasons. Over time the risk of a rover failure increases, hence it is important to get the baseline measurements as quickly as possible. The cost of operating a mission in the traditional manner, with a large co-located science and operations team for 1000 days is very high. In fact, it may not even be possible to obtain sufficient qualified personnel prepared to take time out from their careers to operate a rover for 3 years. Autonomy can alleviate this bottleneck by increasing the rate of science return and decreasing the operations overhead.

Ultimately, to fully explore an area to understand its geology and search for evidence of past or present life may require examining many hundreds, if not thousands, of rocks. Without automation, a few score rocks at most can be examined in a single mission.

Acknowledgements

The authors of this paper would like to acknowledge the contributions of the following people: Randy Sargent, Anne Wright, Susan Lee, Larry Edwards, Clay Kunz, Sailesh Ramakrishnan, Betty Lu, Nicolas Meuleau, Howard Cannon, Hoang Vu, John Bresina, Nathalie Cabrol, and the researchers of the Computational Sciences Division at NASA Ames Research Center, including the Autonomy and Robotics Area managers James Crawford and Nicola Muscettola whose support was pivotal in this endeavor. We would also like to thank the Intelligent Systems and the Mars Technology programs for their support.

References

[1] Bresina, J., R. Dearden, N. Meuleau, S. Ramakrishnan, and R. Washington, "Planning under continuous time and resource uncertainty: a challenge for AI" in *Proc. 19th Conference on Uncertainty in AI*, 2002.

[2] Bresina, J., K. Golden, D. E. Smith, and R. Washington. "Increased flexibility and robustness of Mars rovers," in *Proc. 5th Intl. Symposium on Artificial Intelligence, Robotics and Automation in Space*, 1999.

[3] Bresina, J., and R. Washington. "Robustness via run-time adaptation of contingent plans," in *Proceedings of the AAAI-2001 Spring Symposium: Robust Autonomy*, 2001.

[4] Dearden, R., N. Meuleau, S. Ramakrishnan, D. E. Smith, and R. Washington, "Contingency Planning for Planetary Rovers," in *Proc 3rd NASA International Workshop on Planning and Scheduling for Space*, 2002.

[5] Drummond, M., J. Bresina, and K. Swanson, "Just-In-Case Scheduling," in *Proc. 12th National Conf. on Artificial Intelligence*, 1994.

[6] Frank, J. and A. Jónsson, "Constraint-based attribute and interval planning," to appear in *Constraints*, 2003.

[7] Jónsson, A., P. Morris, N. Muscettola, K. Rajan, B. Smith, "Planning in interplanetary space: theory and practice," in *Proc. 5th Int. Conf. on AI Planning and Scheduling*, 2000, pp. 177–186.

[8] Lawrence, G.M., J.E. Boynton, et al, "CHAMP: Camera HAndlens MicroscoPe," in The 2nd MIDP Conference, Mars Instrument Development Program. JPL Technical Publication D-19508, 2000.

[9] Nesnas, I., R. Volpe, T. Estlin, H. Das, R. Petras, D. Mutz, "Toward Developing Reusable Software Components for Robotic Applications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[10] Nguyen, L., M. Bualat, L. Edwards, L. Flueckiger, C. Neveu, K. Schwehr, M.D. Wagner, and E. Zbinden, "Virtual reality interfaces for visualization and control of remote vehicles," in Autonomous Robots 11(1), 2001.

[11] Pedersen, L., "Science Target Assessment for Mars Rover Instrument Deployment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, September 30 – October 4, 2002.